Ideas for Solving Pandemic

What is the effective time it will take to compute all of the different possibilities for one game?

I estimate there are about half a thousand options in a game. At most not more than 2000 though.

How will I code this?

* The events and the special actions will be the big problem
* Also, detecting game winning/loss states

* It is important that this algorithm is overly efficient so that I can run thousands and thousands of games

* This is gonna have to be multithreaded out the ass
  + I will make a new thread for every turn and return it when they lose
  + I might run out of ram or have stack overflow because I am caching the series of turns probably as a string
    - Maybe output it to a file so that I can do it fine

Building the game:

* I will need to have the game be based on dictionaries, representing the connections between the cities on the board
  + These will allow me to figure out the options available to the player and the amount of cubes present on each square
  + This dictionary will have strings as keys and arrays as values. The array will store:
    - Which player is there, if any
    - How many cubes are there, if any
    - Which cities it is connected to
      * Cities with research stations will be connected
        + Problem with epidemics
* I will also need to have two dictionaries, one for each player
  + This will store the hand of cards
  + This will also maybe store the player's location instead of the dictionary of the locations
    - This way I can query the dictionary for where the player can go next
* I will also need to store the moves that were done during the game

Problems I need to solve:

* How will I enforce the player’s ability to use abilities?
  + Will I implement it as a bunch of disgusting if statements or will I be more graceful?
  + Same for Events?
  + I will have a use <insert thing> function
  + This function will just be the thing that determines whatever the player does. If the player chooses the “move” action, the player should determine what to do in their class
  + Does this imply then that (since I have to iterate and it isn’t just random) there will also be a function of “move” that returns how many different ways they can move so that I can iterate over those possibilities?
    - How much does this actually add to the runtime?

Simple ways to optimize the AI

* Not moving back into a city they were already in this turn unless they have already taken an action other than to move
  + Maybe it would also be efficient to check if they moved in such a way that they could have moved more efficiently?
  + The only way that this would end up being less efficient would be if the runtime of the algorithm every time has a greater runtime than what I gained from just running those extra scenarios
    - I feel like it won't be more efficient unless it is O(n) because just running the other scenarios would be O(n^2)
    - Now that I think about it, it is definitely much more efficient. A lot of the vast options come from moving and there are a lot of ways to move inneficiently
* Maybe I can compare the turns that just ran and only use a fraction of them?
  + How would I objectively weed out games?
    - Didn't use all actions doesn't work
    - Didn't move doesn't work
  + This is likely the most efficient way to go about things

After I have done everything…

* How can I compare all of the different “won” games?
* How will I notice patterns between them that will be meaningful?
  + Perhaps we can also play the games that we saw in the program?
  + Better yet, maybe we play turn by turn so that we can see how many epidemics and cures the top teams have?